МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»

(МТУСИ)

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

по дисциплине

«Одиночное и множественное наследование. Виртуальные и чисто виртуальные функции. Абстрактные и конкретные классы.»

на тему

«Шаблонные функции и классы»

Вариант 15

Выполнил:

студент группы БСУ1801

Матяева В.И.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва, 2021

# Цель работы

Изучить динамические структуры данных, овладеть навыками создания конструкторов копирования, перемещения, деструкторов, перегрузки операций копирования и перемещения.

# Задание

|  |  |  |
| --- | --- | --- |
| int GetSize() const;  void Push(const T &element);  T Pop();  T Peek(); | int GetSize();  void Push(const T &element);  T Pop();  T Peek(); | int GetSize() const;  void PushFront(const T &element);  T PopFront();  void PushBack(const T &element);  T PopBack();  T PeekFront() const;  T PeekBack() const; |

Унаследовать разработанный в предыдущей лабораторной работе класс от созданного в этой работе абстрактного класса.

Разработать ещё одну реализацию динамической структуры данных, указанной в индивидуальном задании. Разработать соответствующий класс, унаследовав его от абстрактного класса и определив все требуемые операции. Предусмотреть конструкторы инициализации, копирования, перемещения, деструктор, функции вставки и удаления элемента, просмотра доступного элемента и функцию, проверяющую наличие элементов.

Перегрузить операции присваивания, перемещения и потокового вывода для вывода содержимого динамической структуры на экран.

Класс разработать в варианте шаблона.

Создать функцию, получающую указатель на базовый класс и демонстрирующую работу

Создать функцию, получающую ссылку на базовый класс и демонстрирующую работу.

**Индивидуальное задание**

**Вариант 15**

|  |  |  |
| --- | --- | --- |
| 15 | Дек на основе двунаправленного циклического списка | DequeBasedOnBidirectionalCyclicLinkedList |

# Выполнение

Листинг 1 – программный код файла Header.h

#include <iostream>

#include "AbstractDeque.h"

using namespace std;

class NoElementsException :public exception

{

public:

NoElementsException(const char\* msg) :

exception(msg) {}

};

class OverflowException :public exception

{

public:

OverflowException(const char\* msg) :

exception(msg) {}

};

template <typename T>

struct Node

{

T data;

Node\* prev, \* next;

};

template <typename T>

class Deque: AbstractDeque<T>

{

public:

Deque();

Deque(T);

Deque& operator=(Deque&& other);

int GetSize();

void pushFront(const T a);

void pushBack(const T a);

void popFront(T& a);

void popBack(T& a);

void peekFront(T& a);

void peekBack(T& a);

private:

Node<T>\* head;

Node<T>\* tail;

int count;

};

Листинг 2 – программный код файла main.cpp

#include "Header.h"

#include <conio.h>

#include<iostream>

using namespace std;

template <typename T>

Deque<T>::Deque()

{

head = tail = nullptr;

count = 0;

}

template <typename T>

Deque<T>::Deque(T a)

{

head = tail = new Node<T>;

head->prev = tail;

head->next = tail;

head->data = a;

}

template <typename T>

Deque<T>& Deque<T>::operator=(Deque&& other)

{

if (this == &other)

{

return \*this;

}

delete head;

head = other.head;

other.head = nullptr;

}

template <typename T>

int Deque<T>::GetSize()

{

return count;

}

template <typename T>

void Deque<T>::pushFront(const T a)

{

Node<T>\* newNode = new Node<T>;

newNode->data = a;

if (head == nullptr)

{

head = tail = newNode;

}

else

{

newNode->next = head;

head->prev = tail->next = newNode;

head = newNode;

}

count++;

};

template <typename T>

void Deque<T>::pushBack(const T a)

{

Node<T>\* newNode = new Node<T>;

newNode->data = a;

if (tail == nullptr)

{

head = tail = newNode;

}

else

{

newNode->prev = tail;

tail->next = head->prev = newNode;

tail = newNode;

}

count++;

};

template <typename T>

void Deque<T>::popFront(T& a)

{

if (head == nullptr)

{

throw NoElementsException("Deck is Empty!");

}

else

{

a = head->data;

Node<T>\* temp = head;

head = head->next;

if (head == nullptr) {

tail = nullptr;

}

else

{

head->prev = tail;

tail->next = head;

}

count--;

}

};

template <typename T>

void Deque<T>::popBack(T& a)

{

if (head == nullptr)

{

throw NoElementsException("Deck is Empty!");

}

else

{

a = tail->data;

Node<T>\* temp = tail;

tail = tail->prev;

if (tail == nullptr) {

head = nullptr;

}

else

{

tail->next = head;

head->prev = tail;

}

count--;

}

};

template <typename T>

void Deque<T>::peekFront(T& a)

{

if (head == nullptr)

{

throw NoElementsException("Deck is Empty!");

}

else

{

a = head->data;

}

}

template <typename T>

void Deque<T>::peekBack(T& a)

{

if (head == nullptr)

{

throw NoElementsException("Deck is Empty!");

}

else

{

a = tail->data;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

int dsize;

double b;

bool flag = true;

cout << "Введите размер дека\n";

cin >> dsize;

Deque<double>\* a;

a = new Deque<double>(dsize);

while (flag)

{

cout << "1. Добавить элемент в начало дека\n";

cout << "2. Получить элемент из начала дека\n";

cout << "3. Добавить элемент в конец дека\n";

cout << "4. Получить элемент из конца дека\n";

cout << "5. Посмотреть элемент в конец дека\n";

cout << "6. Посмотреть элемент из конца дека\n";

cout << "7. Выход\n";

switch (\_getch())

{

case '1':

cout << "Добавить элемент: ";

cin >> b;

a->pushFront(b);

cout << "Элемент добавлен в начало" << endl;

break;

case '2':

try

{

a->popFront(b);

cout << "Полученный элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Дек пуст\n";

}

break;

case '3':

cout << "Добавить элемент: ";

cin >> b;

a->pushBack(b);

cout << "Элемент добавлен в конец " << endl;

break;

case '4':

try

{

a->popBack(b);

cout << "Полученный элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Дек пуст\n";

}

break;

case '5':

try

{

a->peekFront(b);

cout << "Полученный элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Дек пуст\n";

}

break;

case '6':

try

{

a->peekBack(b);

cout << "Полученный элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Дек пуст\n";

}

break;

case '7':

flag = false;

break;

}

}

}

Листинг 3 – AbstractDeque.h

template <typename T>

class AbstractDeque

{

public:

virtual int GetSize() = 0;

virtual void pushFront(const T a) = 0;

virtual void pushBack(const T a) = 0;

virtual void popFront(T& a) = 0;

virtual void popBack(T& a) = 0;

virtual void peekFront(T& a) = 0;

virtual void peekBack(T& a) = 0;

};

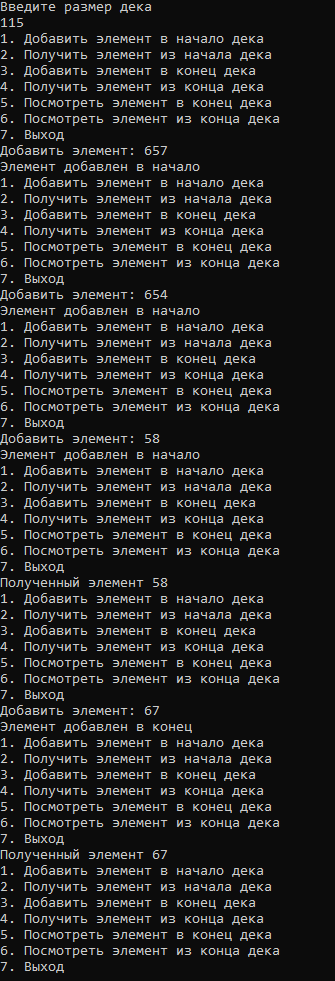


Рисунок 1 – Результат работы с программой.